

Un petit langage fonctionnel : FUN.

On se rapproche de notre but final en considérant un petit langage fonctionnel, nommé FUN.

On se donne l'ensemble des entiers relatifs \mathbb{Z} et un ensemble infini de variables \mathcal{V} . L'ensemble des expressions de FUN, notées e , e' ou e_i , est défini par la grammaire suivante :

$$e ::= k \mid e_1 + e_2 \mid \underbrace{\text{fun } x \rightarrow e}_{\text{Fonction / Abstraction}} \mid \overbrace{e_1 \ e_2}^{\text{Application}} \mid x.$$

Note 1. On simplifie la notation par rapport à EA ou LEA : on ne souligne plus les entiers, on n'entoure plus les plus.

On notera de plus $e_1 \ e_2 \ e_3$ pour $(e_1 \ e_2) \ e_3$. Aussi, l'expression $\text{fun } x \ y \rightarrow e$ représentera l'expression $\text{fun } x \rightarrow (\text{fun } y \rightarrow e)$. On n'a pas le droit à plusieurs arguments pour une fonction, mais on applique la curryfication.

1 Sémantique opérationnelle « informellement ».

Exemple 1. Comment s'évalue $(\text{fun } x \rightarrow x + x)(7 + 7)$?

- ▷ D'une part, $7 + 7$ s'évalue en 14.
- ▷ D'autre part, $(\text{fun } x \rightarrow x + x)$ s'évalue en elle même.

- ▷ On procède à une substitution de $(x + x)[14/x]$ qui s'évalue en 28.

Exemple 2. Comment s'évalue l'expression

$$\overbrace{((\text{fun } f \rightarrow (\text{fun } x \rightarrow x + (f \ x))) (\text{fun } y \rightarrow y + y))}^A \ 7 ?$$

$\underbrace{\hspace{10em}}_B \qquad \underbrace{\hspace{10em}}_C$

On commence par évaluer A et C qui s'évaluent en A et C respectivement. On continue en calculant la substitution

$$(\text{fun } x \rightarrow x + (f \ x))[(\text{fun } y \rightarrow y + y/f)],$$

ce qui donne

$$(\text{fun } x \rightarrow x + ((\text{fun } y \rightarrow y + y) \ x)).$$

Là, on **ne simplifie pas**, car c'est du code *dans* une fonction. On calcule ensuite la substitution

$$(x + ((\text{fun } y \rightarrow y + y) \ x))[7/x],$$

ce qui donne

$$7 + ((\text{fun } y \rightarrow y + y) \ 7).$$

On termine par la substitution

$$(y + y)[7/y] = 7 + 7.$$

On conclut que l'expression originelle s'évalue en 21.

Remarque 1. Dans FUN, le résultat d'un calcul (qu'on appellera *valeur*) n'est plus forcément un entier, ça peut aussi être une fonction.

L'ensemble des valeurs, notées v , est défini par la grammaire

$$v ::= k \mid \mathbf{fun} \ x \rightarrow e.$$

LES FONCTIONS SONT DES VALEURS ! Et, le « contenu » la fonction n'est pas forcément une valeur.

On peut remarquer que l'ensemble des valeurs est un sous-ensemble des expressions de FUN.

2 Sémantique opérationnelle de FUN (version 1).

Définition 1. On définit l'ensemble des *variables libres* $\mathcal{V}\ell(e)$ d'une expression e par (on a 5 cas) :

- ▷ $\mathcal{V}\ell(x) = \{x\}$;
- ▷ $\mathcal{V}\ell(k) = \emptyset$;
- ▷ $\mathcal{V}\ell(e_1 + e_2) = \mathcal{V}\ell(e_1) \cup \mathcal{V}\ell(e_2)$;
- ▷ $\mathcal{V}\ell(e_1 \ e_2) = \mathcal{V}\ell(e_1) \cup \mathcal{V}\ell(e_2)$;
- ▷ $\mathcal{V}\ell(\mathbf{fun} \ x \rightarrow e) = \mathcal{V}\ell(e) \setminus \{x\}$.¹

On dit que e est *close* si $\mathcal{V}\ell(e) = \emptyset$.

Définition 2. Pour $e \in \mathbf{FUN}$, $x \in \mathcal{V}$ et v une valeur **close**, on définit la *substitution* $e[v/x]$ de x par v dans e par :

- ▷ $k[v/x] = k$;
- ▷ $y[v/x] = \begin{cases} v & \text{si } x = y \\ y & \text{si } x \neq y \end{cases}$;
- ▷ $(\mathbf{fun} \ y \rightarrow e)[v/x] = \begin{cases} \mathbf{fun} \ y \rightarrow e & \text{si } x = y \\ \mathbf{fun} \ y \rightarrow e[e/x] & \text{si } x \neq y \end{cases}$;
- ▷ $(e_1 + e_2)[v/x] = (e_1[v/x]) + (e_2[v/x])$;
- ▷ $(e_1 \ e_2)[v/x] = (e_1[v/x]) \ (e_2[v/x])$.

1. L'expression $\mathbf{fun} \ x \rightarrow e$ est un *lieur* : x est liée dans e .

2.1 Grands pas pour FUN.

On définit la relation \Downarrow sur couples (expression, valeur) par :

$$\begin{array}{c}
 k = k_1 + k_2 \quad \frac{e_1 \Downarrow k_1 \quad e_2 \Downarrow k_2}{e_1 + e_2 \Downarrow k} \quad \frac{}{v \Downarrow v} \\
 \\
 \frac{e_1 \Downarrow \text{fun } x \rightarrow e \quad e_2 \Downarrow v_2 \quad e[v_2/x] \Downarrow v}{e_1 e_2 \Downarrow v.}
 \end{array}$$

Remarque 2. Certaines expressions ne s'évaluent pas :

$$x \not\Downarrow \quad \text{et} \quad z + (\text{fun } x \rightarrow x) \not\Downarrow$$

par exemple.

2.2 Petits pas pour FUN.

On définit la relation $\rightarrow \subseteq \text{FUN} * \text{FUN}$ par :

$$\begin{array}{c}
 k = k_1 + k_2 \quad \frac{}{k_1 + k_2 \rightarrow k} \mathcal{R}_{\text{pk}} \quad \frac{}{(\text{fun } x \rightarrow e) v \rightarrow e[v/x]} \mathcal{R}_{\beta} \\
 \\
 \frac{e_2 \rightarrow e'_2}{e_1 + e_2 \rightarrow e_1 + e'_2} \mathcal{R}_{\text{pd}} \quad \frac{e_1 \rightarrow e'_1}{e_1 + k \rightarrow e'_1 + k} \mathcal{R}_{\text{pg}} \\
 \\
 \frac{e_2 \rightarrow e'_2}{e_1 e_2 \rightarrow e_1 e'_2} \mathcal{R}_{\text{ad}} \quad \frac{e_1 \rightarrow e'_1}{e_1 v \rightarrow e'_1 v} \mathcal{R}_{\text{ag}}.
 \end{array}$$

Remarque 3. Il existe des expressions que l'on ne peut pas réduire :

1. $k \not\rightarrow$;
2. $(\text{fun } x \rightarrow x) \not\rightarrow$;
3. $e_1 + (\text{fun } x \rightarrow x) \not\rightarrow$;
4. $3 (5 + 7) \rightarrow 3 12 \not\rightarrow$.

Dans les cas 1. et 2., c'est cohérent : on ne peut pas réduire des valeurs.

Lemme 1. On a

$$e \Downarrow v \quad \text{si, et seulement si,} \quad e \rightarrow^* v.$$

Remarque 4. Soit $e_0 = (\text{fun } x \rightarrow x \ x) \ (\text{fun } x \rightarrow x \ x)$. On remarque que $e_0 \rightarrow e_0$.

En FUN, il y a des divergences : il existe $(e_n)_{n \in \mathbb{N}}$ telle que l'on ait $e_n \rightarrow e_{n+1}$.

La fonction² définie par \Downarrow est donc partielle.

Remarque 5 (Problème avec la substitution). On a la chaîne de réductions :

$$\begin{aligned} & ((\text{fun } y \rightarrow (\text{fun } x \rightarrow x + y)) \ (x + 7)) \ 5 \\ (\star) \quad & \rightarrow (\text{fun } x \rightarrow x + (x + 7)) \ 5 \\ & \rightarrow 5 + (5 + 7) \\ & \rightarrow^* 17. \end{aligned}$$

Attention ! Ici, on a triché : on a substitué avec l'expression $x + 7$ mais ce n'est pas une valeur (dans la réduction (\star)) !

Mais, on a la chaîne de réductions

$$\begin{aligned} & (\text{fun } f \rightarrow (\text{fun } x \rightarrow (f \ 3) + x)) \ (\text{fun } t \rightarrow x + 7) \ 5 \\ & \rightarrow (\text{fun } x \rightarrow ((\text{fun } t \rightarrow x + 7) \ 3) + x) \ 5 \\ & \rightarrow (\text{fun } x \rightarrow ((\text{fun } t \rightarrow x + 7) \ 3) + x) \ 5. \end{aligned}$$

Et là, c'est le drame, on a **capturé la variable libre**. D'où l'hypothèse de v close dans la substitution.

2. Pour indiquer cela, il faudrait démontrer que la relation \Downarrow est déterministe.

Remarque 6. Les relations \Downarrow et \rightarrow sont définies sur des expressions **closer**. Et on a même $\rightarrow \subseteq \text{FUN}_0 * \text{FUN}_0$.³

Lemme 2. \triangleright Si v est close et si $x \notin \mathcal{V}\ell(e)$ alors $e[v/x] = e$.

\triangleright Si v est close, $\mathcal{V}\ell(e[v/x]) = \mathcal{V}\ell(e) \setminus \{x\}$. \square

Lemme 3. Si $e \in \text{FUN}_0$ et $e \rightarrow e'$ alors $e' \in \text{FUN}_0$.

Preuve. Montrons que, quelles que soient e et e' , on a : si $e \rightarrow e'$ alors $(e \in \text{FUN}_0) \implies (e' \in \text{FUN}_0)$ On procède par induction sur la relation $e \rightarrow e'$. Il y a 6 cas :

1. Pour \mathcal{R}_β , on suppose $(\text{fun } x \rightarrow e) v$ est close, alors

\triangleright $(\text{fun } x \rightarrow e)$ est close ;

\triangleright v est close.

On sait donc que $\mathcal{V}\ell(e) \subseteq \{x\}$, d'où par le lemme précédent, $\mathcal{V}\ell(e[v/x]) = \emptyset$ et donc $e[v/x]$ est close.

2–6. Pour les autres cas, on procède de la même manière. \square

Remarque 7. De même, si $e \Downarrow v$ où e est close, alors v est close.

Les relations \Downarrow et \rightarrow sont définies sur les expressions et les valeurs closes.

Définition 3 (Définition informelle de l' α -conversion). On définit l' α -conversion, notée $e =_\alpha e'$: on a $\text{fun } x \rightarrow e =_\alpha \text{fun } y \rightarrow e'$ si, et seulement si, e' s'obtient en remplaçant x par y dans e à condition que $y \notin \mathcal{V}\ell(e)$.⁴

On étend $e =_\alpha e'$ à toutes les expressions : « on peut faire ça

3. Il faudrait ici justifier que la réduction d'une formule close est close. C'est ce que nous allons justifier.

partout ».

Exemple 3 (*Les variables liées sont muettes.*). On a :

$$\begin{aligned} \text{fun } x \rightarrow x + z &=_{\alpha} \text{fun } y \rightarrow y + z \\ &=_{\alpha} \text{fun } t \rightarrow t + z \\ &\neq_{\alpha} \text{fun } z \rightarrow z + z. \end{aligned}$$

L'intuition est, quand on a $\text{fun } x \rightarrow e$ et qu'on a besoin de renommer la variable x , pour cela on prend $x' \notin \mathcal{V}\ell(e)$.

“Lemme” 1. Si $E_0 \subseteq \mathcal{V}$ est un ensemble fini de variables, alors il existe $z \notin E_0$ et $e' \in \text{FUN}$ tel que $\text{fun } x \rightarrow e =_{\alpha} \text{fun } z \rightarrow e'$. \square

Remarque 8 (Fondamental). En fait FUN désigne l'ensemble des expressions décrites par la grammaire initiale *quotientée* par α -conversion.

Remarque 9. On remarque que

$$(e =_{\alpha} e') \implies \mathcal{V}\ell(e) = \mathcal{V}\ell(e').$$

D'après le “lemme”, on peut améliorer notre définition de la substitution.

Définition 4. Pour $e \in \text{FUN}$, $x \in \mathcal{V}$ et v une valeur **close**, on définit la *substitution* $e[v/x]$ de x par v dans e par :

- ▷ $k[v/x] = k$;
- ▷ $y[v/x] = \begin{cases} v & \text{si } x = y \\ y & \text{si } x \neq y ; \end{cases}$
- ▷ $(\text{fun } x \rightarrow e)[v/x] = (\text{fun } y \rightarrow e)[v/x]$ lorsque $x \neq y$;

4. C'est une « variable fraîche ».

- ▷ $(e_1 + e_2)[v/x] = (e_1[v/x]) + (e_2[v/x])$;
- ▷ $(e_1 e_2)[v/x] = (e_1[v/x]) (e_2[v/x])$.

3 Ajout des déclarations locales (FUN + let).

On ajoute les déclarations locales (comme pour EA → LEA) à notre petit langage fonctionnel. Dans la grammaire des expressions de FUN, on ajoute :

$$e ::= \dots \mid \text{let } x = e_1 \text{ in } e_2.$$

Ceci implique d'ajouter quelques éléments aux différentes opérations sur les expressions définies ci-avant :

- ▷ on définit $\mathcal{Vl}(\text{let } x = e_1 \text{ in } e_2) = \mathcal{Vl}(e_1) \cup (\mathcal{Vl}(e_2) \setminus \{x\})$;
- ▷ on ne change pas les valeurs : une déclaration locale n'est pas une valeur ;
- ▷ on ajoute $\text{let } x = e_1 \text{ in } e_2 =_\alpha \text{let } y = e_1 \text{ in } e'_2$, où l'on remplace x par y dans e_2 pour obtenir e'_2 ;
- ▷ pour la substitution, on pose lorsque $x \neq y$ (que l'on peut toujours supposer modulo α -conversion)

$$(\text{let } y = e_1 \text{ in } e_2)[v/x] = (\text{let } y = e_1[v/x] \text{ in } e_2[v/x]).$$

- ▷ pour la sémantique à grands pas, c'est comme pour LEA ;
- ▷ pour la sémantique à petits pas, on ajoute les deux règles :

$$\frac{}{\text{let } x = v \text{ in } e_2 \rightarrow e_2[v/x]} \mathcal{R}_{lv}$$

et

$$\frac{e_1 \rightarrow e'_1}{\text{let } x = e_1 \text{ in } e_2 \rightarrow \text{let } x = e'_1 \text{ in } e_2} \mathcal{R}_{lg}.$$

Attention! On n'a pas de règle

~~$$\frac{e_2 \rightarrow e'_2}{\text{let } x = e_1 \text{ in } e_2 \rightarrow \text{let } x = e_1 \text{ in } e'_2} \mathcal{R}_{ld},$$~~

on réduit d'abord l'expression e_1 jusqu'à une valeur, avant de passer à e_2 .

Le langage que l'on construit s'appelle FUN + let.

3.1 Traduction de FUN + let vers FUN.

On définit une fonction qui, à toute expression de e dans FUN + let associe une expression notée $\llbracket e \rrbracket$ dans FUN (on supprime les expressions locales). L'expression $\llbracket e \rrbracket$ est définie par induction sur e . Il y a 6 cas :

- ▷ $\llbracket k \rrbracket = k$;
- ▷ $\llbracket x \rrbracket = x$;
- ▷ $\llbracket e_1 + e_2 \rrbracket = \llbracket e \rrbracket_1 + \llbracket e \rrbracket_2$;
- ▷ $\llbracket e_1 e_2 \rrbracket = \llbracket e \rrbracket_1 \llbracket e \rrbracket_2$;
- ▷ $\llbracket \text{fun } x \rightarrow e \rrbracket = \text{fun } x \rightarrow \llbracket e \rrbracket$;
- ▷ $\llbracket \text{let } x = e_1 \text{ in } e_2 \rrbracket = (\text{fun } x \rightarrow \llbracket e_2 \rrbracket) \llbracket e_1 \rrbracket$.

Lemme 4. Pour tout $e \in (\text{FUN} + \text{let})$,

- ▷ $\llbracket e \rrbracket$ est une expression de FUN⁵ ;
- ▷ on a $\mathcal{V}\ell(\llbracket e \rrbracket) = \mathcal{V}\ell(e)$;
- ▷ $\llbracket e \rrbracket$ est une valeur ssi e est une valeur ;
- ▷ $\llbracket e[v/x] \rrbracket = \llbracket e \rrbracket [\llbracket v \rrbracket/x]$ ⁶. □

Pour démontrer le lemme 4, on procède par induction sur e . C'est long et rébarbatif, mais la proposition ci-dessous est bien plus intéressante.

Proposition 1. Pour toutes expressions e, e' de FUN + let, si on a la réduction $e \rightarrow_{\text{FUN} + \text{let}} e'$ alors $\llbracket e \rrbracket \rightarrow_{\text{FUN}} \llbracket e' \rrbracket$.

Preuve. On procède par induction sur $e \rightarrow e'$ dans FUN + let. Il y a 8 cas car il y a 8 règles d'inférences pour \rightarrow dans FUN + let.

- ▷ Cas \mathcal{R}_{lv} . Il faut montrer que $\llbracket \text{let } x = v \text{ in } e_2 \rrbracket \rightarrow_{\text{FUN}} \llbracket e[v/x] \rrbracket$. Par définition, l'expression de droite vaut

$$(\text{fun } x \rightarrow \llbracket e \rrbracket_2) \llbracket v \rrbracket \xrightarrow{\mathcal{R}_\beta}_{\text{FUN}} \llbracket e \rrbracket_2 [\llbracket v \rrbracket/x],$$

5. i.e. $\llbracket e \rrbracket$ n'a pas de déclarations locales

6. On le prouve par induction sur e , c'est une induction à 6 cas

car $\llbracket v \rrbracket$ est une valeur par le lemme 4, ce qui justifie \mathcal{R}_β . De plus, encore par le lemme 4, on a l'égalité entre $\llbracket e \rrbracket_2 \llbracket \llbracket v \rrbracket / x \rrbracket = \llbracket e[v/x] \rrbracket$.

- ▷ *Cas \mathcal{R}_{lg} .* On sait que $e_1 \rightarrow e'_1$ et, par hypothèse d'induction, on a $\llbracket e_1 \rrbracket \rightarrow \llbracket e'_1 \rrbracket$. Il faut montrer que

$$\llbracket \text{let } x = e_1 \text{ in } e_2 \rrbracket \rightarrow \llbracket \text{let } x = e'_1 \text{ in } e_2 \rrbracket.$$

L'expression de droite vaut

$$(\text{fun } x \rightarrow \llbracket e_2 \rrbracket) \llbracket e_1 \rrbracket \xrightarrow{\mathcal{R}_{ad} \ \& \ \text{hyp. ind.}} (\text{fun } x \rightarrow \llbracket e_2 \rrbracket) \llbracket e'_1 \rrbracket.$$

Et, par définition de $\llbracket \cdot \rrbracket$, on a l'égalité :

$$\llbracket \text{let } x = e'_1 \text{ in } e_2 \rrbracket = (\text{fun } x \rightarrow \llbracket e_2 \rrbracket) \llbracket e'_1 \rrbracket.$$

- ▷ Les autres cas sont laissées en exercice. □

Proposition 2. Si $\llbracket e \rrbracket \rightarrow \llbracket e' \rrbracket$ alors $e \rightarrow e'$.

Preuve. La proposition ci-dessus est mal formulée pour être prouvée par induction, on la ré-écrit. On démontre, par induction sur la relation $f \rightarrow f'$ la propriété suivante :

« quel que soit e , si $f = \llbracket e \rrbracket$ alors il existe e' une expression telle que $f' = \llbracket e' \rrbracket$ et $e \rightarrow e'$ (dans FUN + let) »,

qu'on notera $\mathcal{P}(f, f')$.

Pour l'induction sur $f \rightarrow f'$, il y a 6 cas.

- ▷ *Cas de la règle \mathcal{R}_{ad} .* On suppose $f_2 \rightarrow f'_2$ et par hypothèse d'induction $\mathcal{P}(f_2, f'_2)$. On doit montrer $\mathcal{P}(f_1 \ f_2, f_1 \ f'_2)$. On suppose donc $\llbracket e \rrbracket = f_1 \ f_2$. On a deux sous-cas.
- *1^{er} sous-cas.* On suppose $e = e_1 \ e_2$ et $\llbracket e_1 \rrbracket = f_1 = f_2$. Par hypothèse d'induction et puisque $\llbracket e_2 \rrbracket = f_2$, il

existe e'_2 tel que $e_2 \rightarrow e'_2$ et $\llbracket e'_2 \rrbracket = f'_2$. De $e_2 \rightarrow e'_2$, on en déduit par \mathcal{R}_{ad} que $e_1 e_2 \rightarrow e_1 e'_2$. On pose $e' = e_1 e'_2$ et on a bien $\llbracket e' \rrbracket = \llbracket e_1 \rrbracket \llbracket e'_2 \rrbracket$.

- 2^{ème} sous-cas. On suppose $e = \text{let } x = e_1 \text{ in } e_2$. Alors,

$$\llbracket e \rrbracket = \underbrace{(\text{fun } x \rightarrow \llbracket e_2 \rrbracket)}_{f_1} \underbrace{\llbracket e_1 \rrbracket}_{f_2}.$$

Par hypothèse d'induction, il existe e'_1 tel que $e_1 \rightarrow e'_1$ et $\llbracket e'_1 \rrbracket = f'_2$. Posons $e' = (\text{let } x = e'_1 \text{ in } e_2)$. On doit vérifier $\llbracket e \rrbracket \rightarrow \llbracket e' \rrbracket$ ce qui est vrai par \mathcal{R}_{ad} et que $\llbracket e' \rrbracket = f_1 f'_2$, ce qui est vrai par définition.

- ▷ Cas de la règle \mathcal{R}_{ag} . On suppose $f_1 \rightarrow f'_1$ et l'hypothèse d'induction $\mathcal{P}(f_1, f'_1)$. On doit vérifier que $\mathcal{P}(f_1 v, f'_1 v)$. On suppose $\llbracket e \rrbracket = f_1 v$ et on a deux sous-cas.
 - 1^{er} sous-cas. On suppose $e = e_1 e_2$ et alors $\llbracket e \rrbracket = \llbracket e_1 \rrbracket \llbracket e_2 \rrbracket$ par le lemme 4 et parce que e_2 est une valeur (car $\llbracket e_2 \rrbracket = v$). On raisonne comme pour la règle \mathcal{R}_{ad} dans le premier sous-cas, en appliquant \mathcal{R}_{ag} .
 - 2nd sous-cas. On suppose $e = (\text{let } x = e_1 \text{ in } e_2)$ alors

$$\llbracket e \rrbracket = \underbrace{\text{fun } x \rightarrow \llbracket e_2 \rrbracket}_{f_1} \underbrace{\llbracket e_1 \rrbracket}_{f_2}.$$

On vérifie aisément ce que l'on doit montrer.

- ▷ Les autres cas se font de la même manière (attention à \mathcal{R}_{β}).

□